
HIPS 2013 Conference

# How to Scale Dynamic Tuning to Large Parallel Applications

**Andrea Martínez, Anna Sikora, Eduardo César, Joan Sorribes**

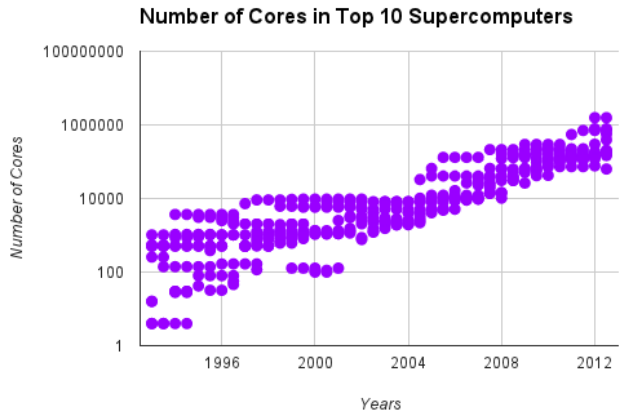
Computer Architecture and Operating Systems Department
Universidad Autónoma de Barcelona

Universitat Autònoma de Barcelona

# Outline

- Motivation.
- Large-scale Dynamic Tuning.
- Scalability Evaluation.
- Conclusions and Future Work.

# Motivation




# Performance ?

## PERFORMANCE ANALYSIS TOOLS

CHALLENGES

**Dynamic tuning**

- Active Harmony
- MATE

**Scalability**

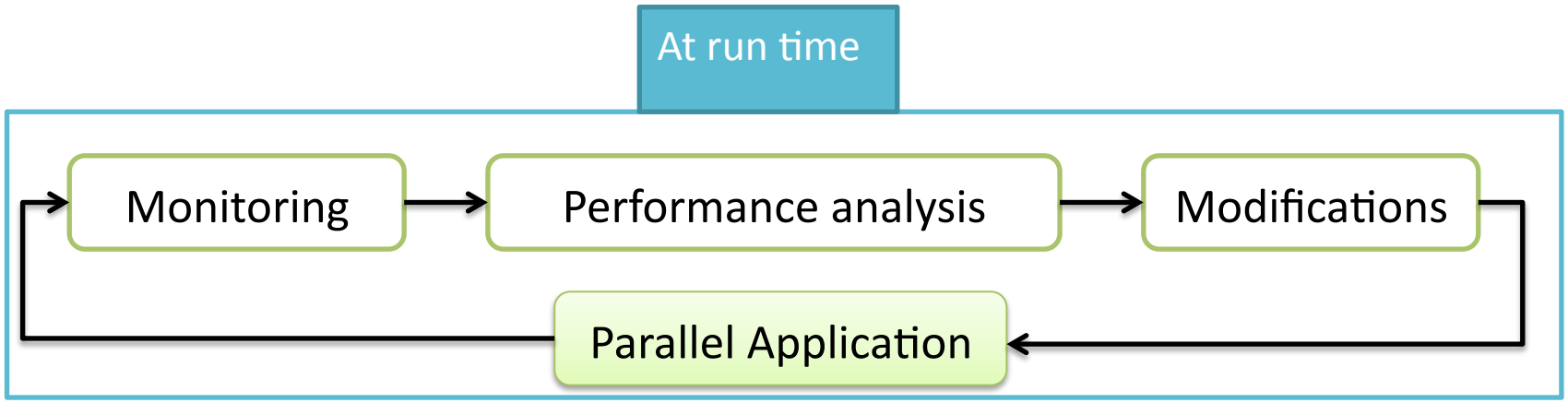
- Periscope
- Scalasca

?

# Motivation

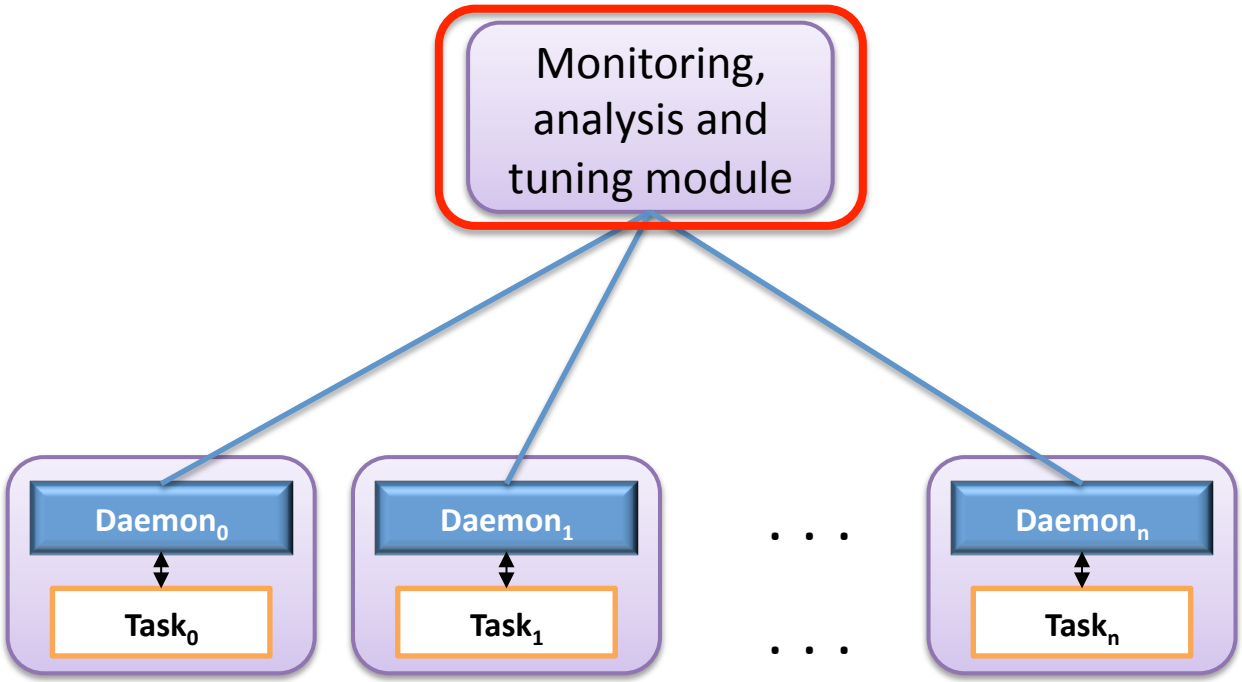
Define an approach that provides automatic and dynamic analysis and tuning of large-scale parallel applications

## DYNAMIC TUNING



# Motivation

Define an approach that provides automatic and dynamic analysis and tuning of large-scale parallel applications



# Outline

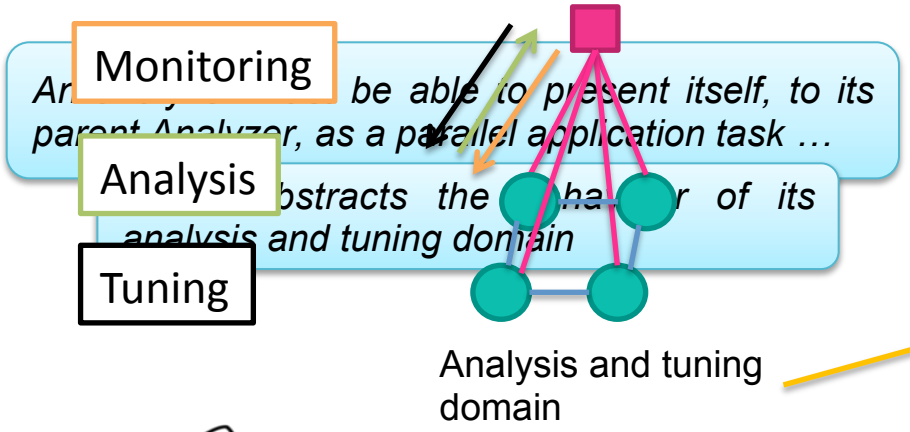
- Motivation.
- Large-scale Dynamic Tuning.
- Scalability Evaluation.
- Conclusions and future work.

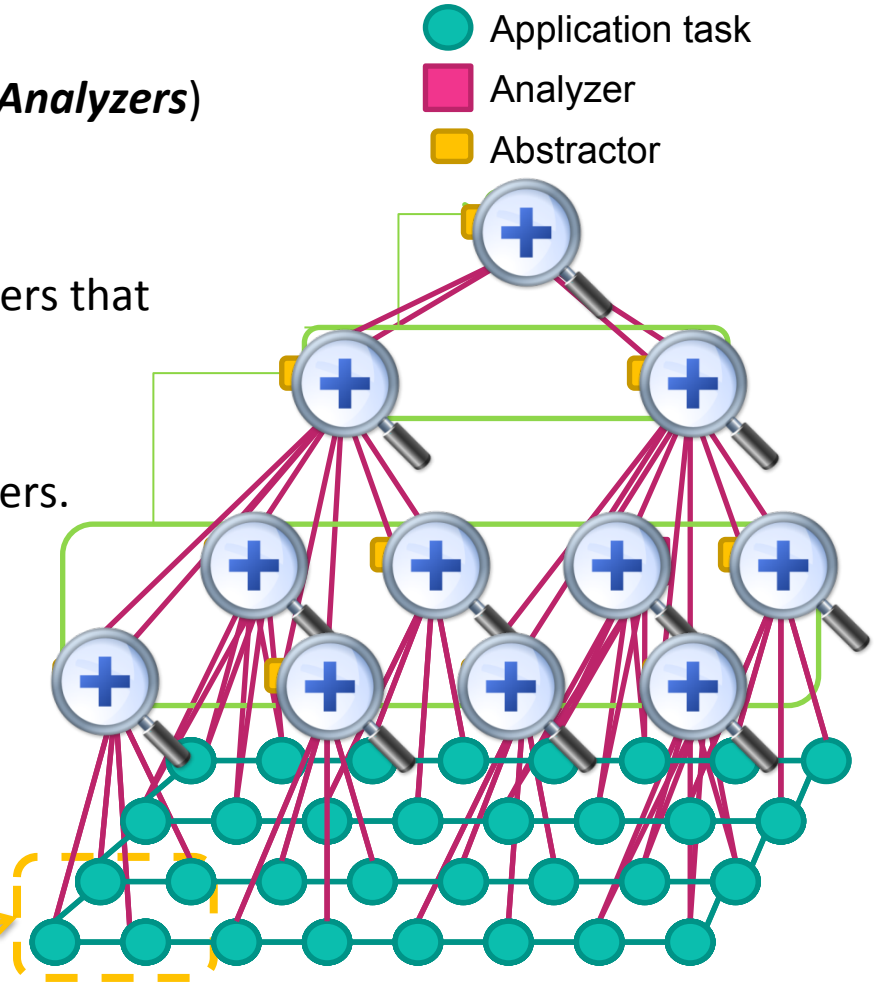
# Hierarchical Tuning Network

- It is composed of **analysis and tuning modules** (***Analyzers***) structured as a hierarchical tree.

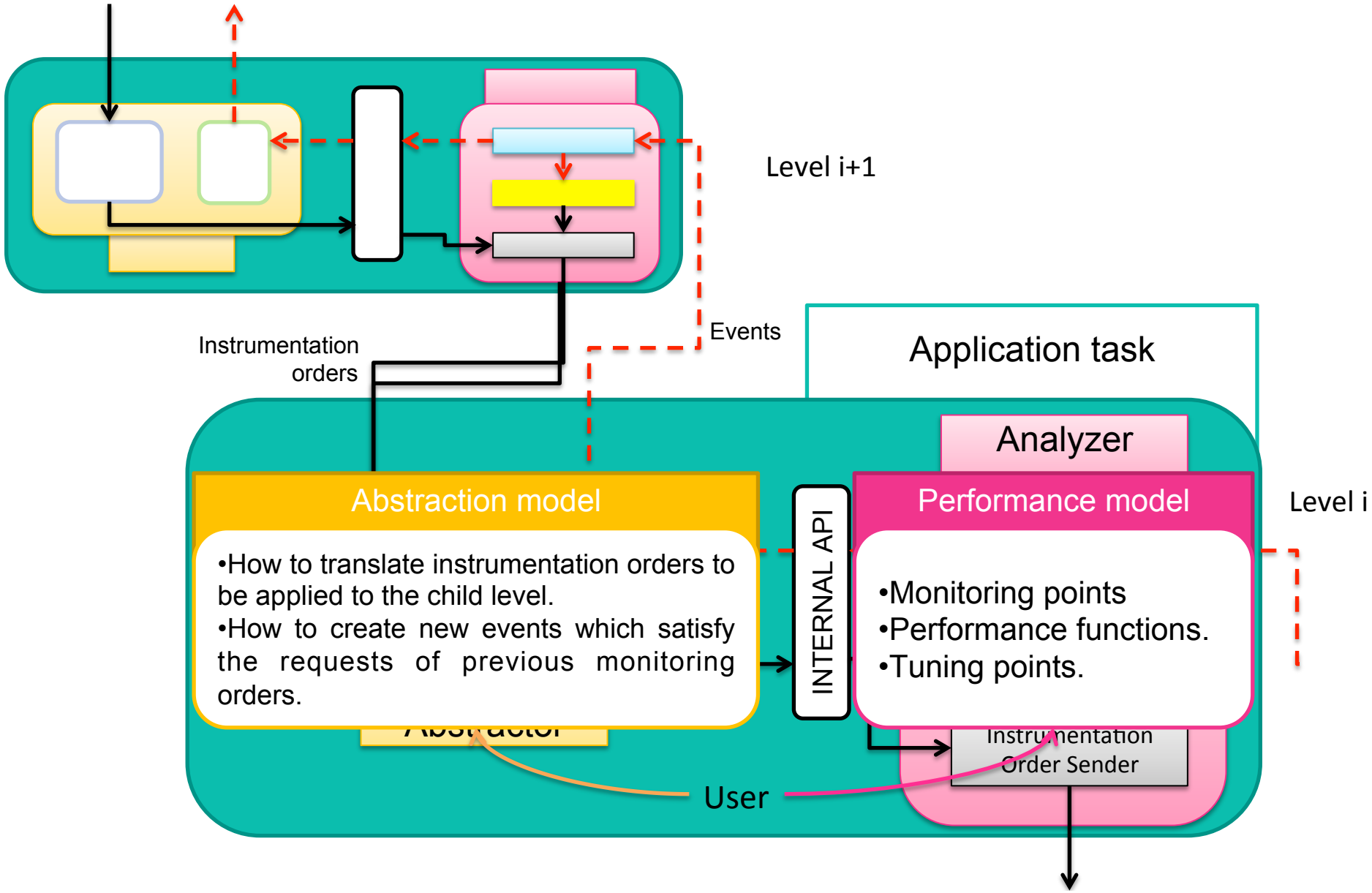
- The base of the hierarchy is composed of Analyzers that controls disjoint subsets of application tasks

- Abstraction mechanism between levels of Analyzers.

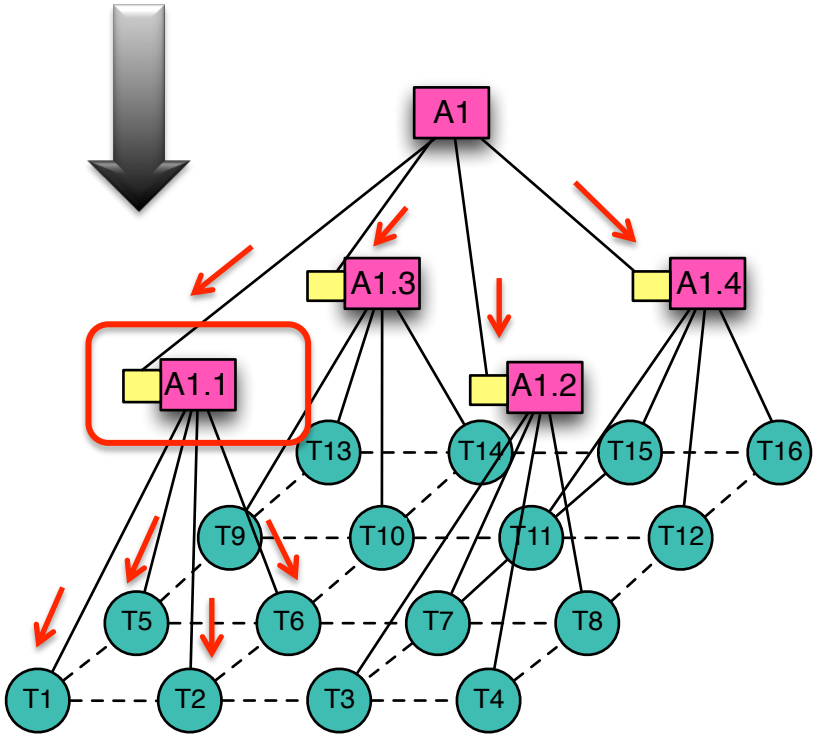


# Abstraction mechanism



# Abstraction example




**Monitoring order translation**

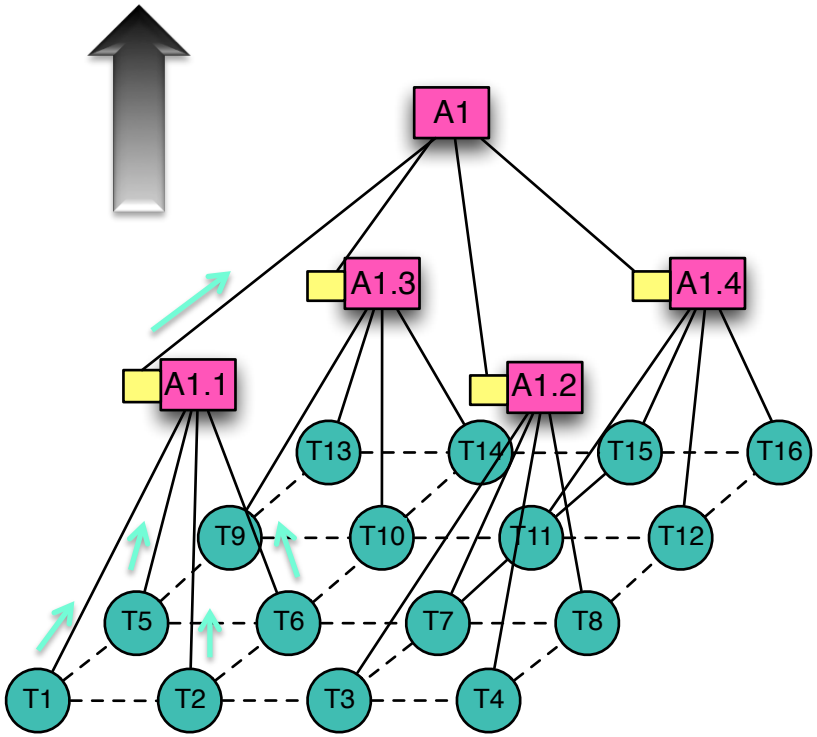
A1 generates:

- **Mon_Order**(computation_time, #work_units) to A1.1
- **Mon_Order**(computation_time, #work_units) to A1.2
- **Mon_Order**(computation_time, #work_units) to A1.3
- **Mon_Order**(computation_time, #work_units) to A1.4

Abstractor associated to A1.1 generates:

- **Mon_Order**(computation_time, #work_units) to T1
- **Mon_Order**(computation_time, #work_units) to T2
- **Mon_Order**(computation_time, #work_units) to T5
- **Mon_Order**(computation_time, #work_units) to T6.

# Abstraction example




**Event Creation**

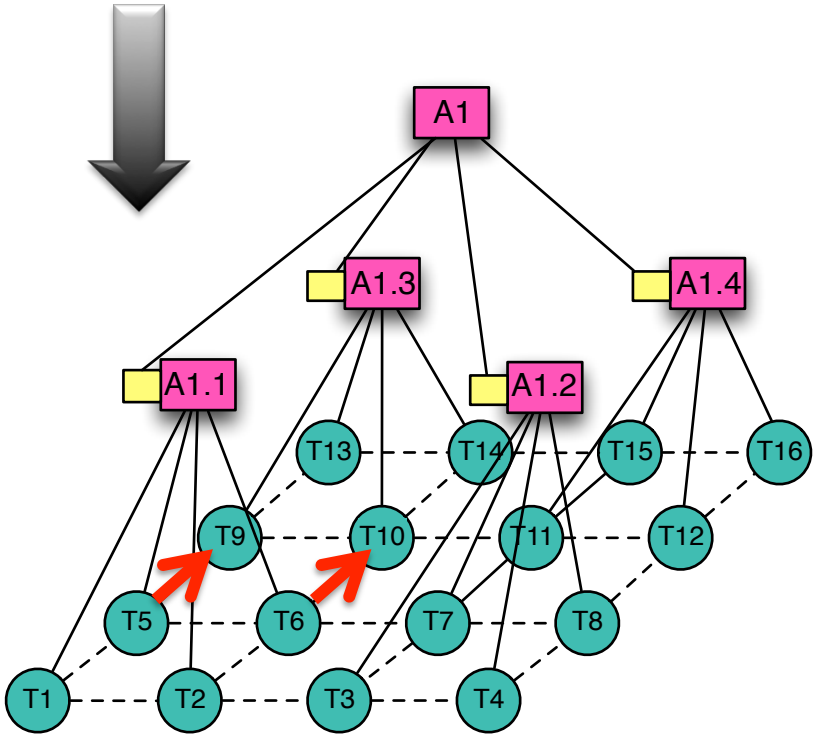
A1.1 receives:

- **Event**(computation_time, #work_units) from T1
- **Event**(computation_time, #work_units) from T2
- **Event**(computation_time, #work_units) from T5
- **Event**(computation_time, #work_units) from T6

Using the abstraction model, the Abstractor associated to A1.1 creates a new event:

- **Event**(Mean(computation_time), Sum(#work_units)) to A1

# Abstraction example




**Tuning order translation**

A1 generates:

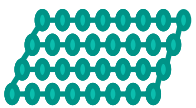
- **TuningOrder**(send_load, to_A1.3, #work_units) to A1.1

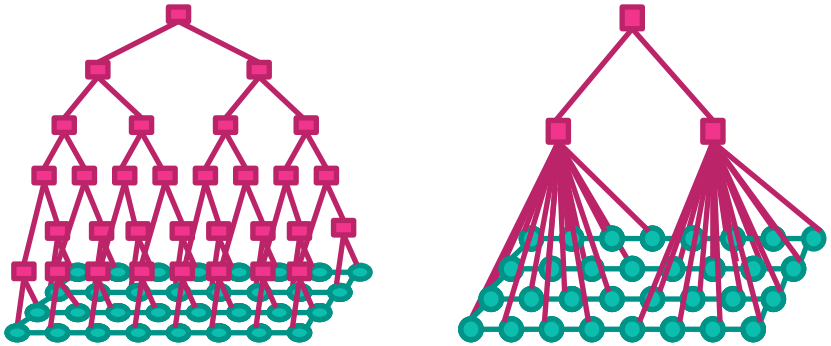
Using the abstraction model, the Abstractor associated to A1.1 generates:

- **TuningOrder**(send_load, to_T9, #work_units/2) to T5
- **TuningOrder**(send_load, to_T10, #work_units/2) to T6

# Determining the architecture topology

Parallel application of N tasks





The use of architecture topologies with the minimum number of non-saturated analysis modules.

Work during the analysis and tuning process

$$N \cdot E_a \cdot T_m + T_a(N) + \frac{E_a}{E_c} \cdot T_c + T_t \cdot f_{rp}$$

Gathering performance data

Performance analysis

Event creation

Order translation

# Outline

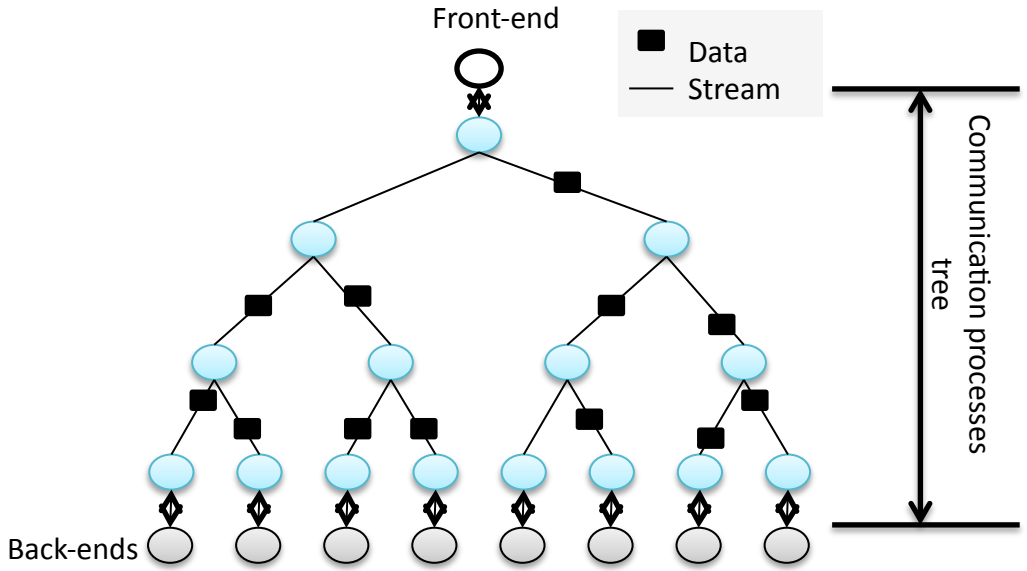
- Motivation.
- Large-scale Dynamic Tuning.
- Scalability evaluation.
- Conclusions and future work.

# Scalability Evaluation

- Prototype implementation of the proposed tuning network

## MRNet Framework

- •A network of hierarchically organised processes .
- •Potential: Filters.




Functionality
Abstractor-Analyzer → Filter

The prototype simulates all the actions which would take place during the performance analysis and tuning process of a parallel application:

- BEs simulate an instrumented SPMD application.
- Filters simulate the Abstractor-Analyzer functionality.

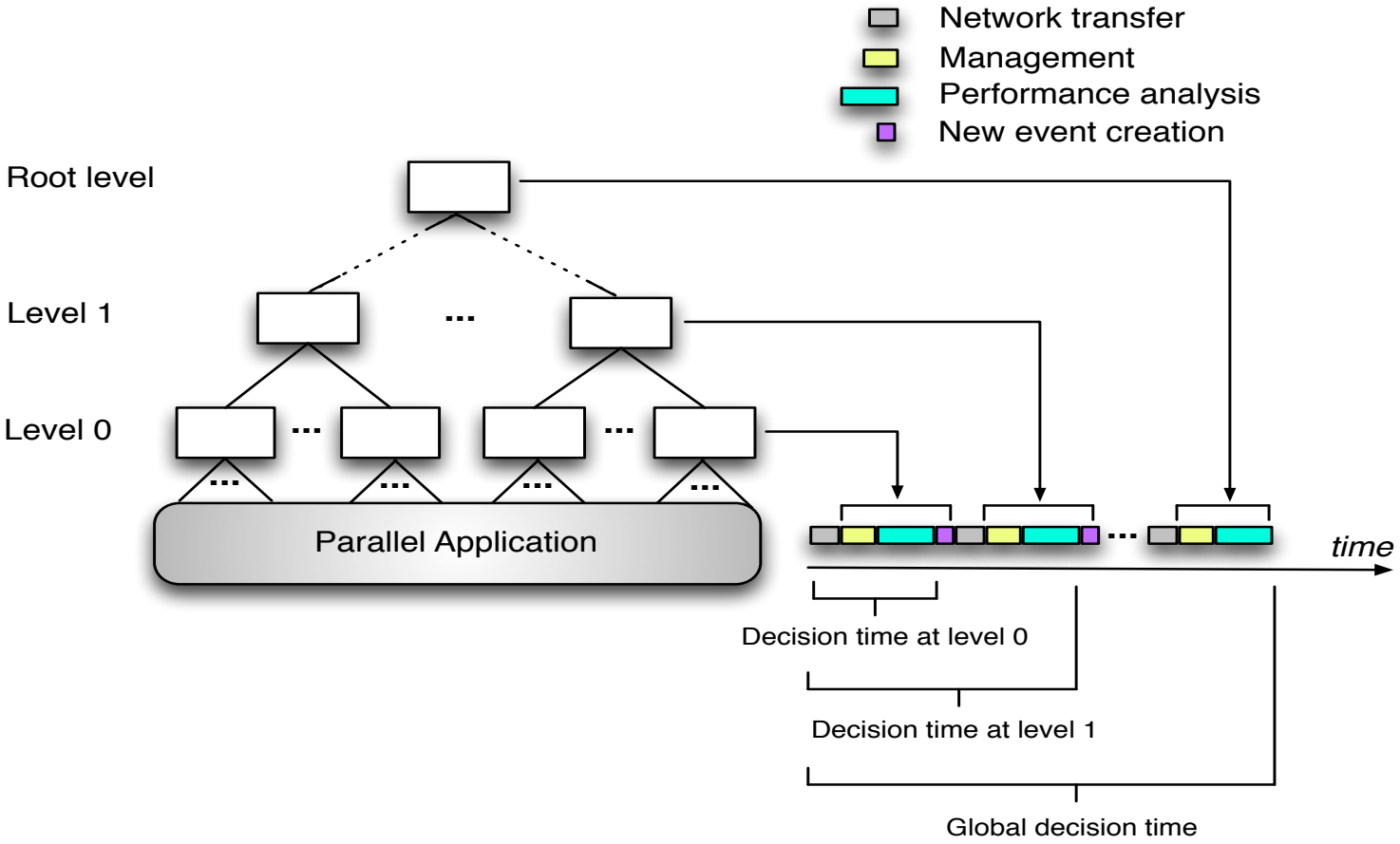
# Scalability Evaluation

- Execution Environments:
  - Marenostrum at Barcelona Supercomputing Centre
  - SuperMUC at Leibniz Supercomputing Centre

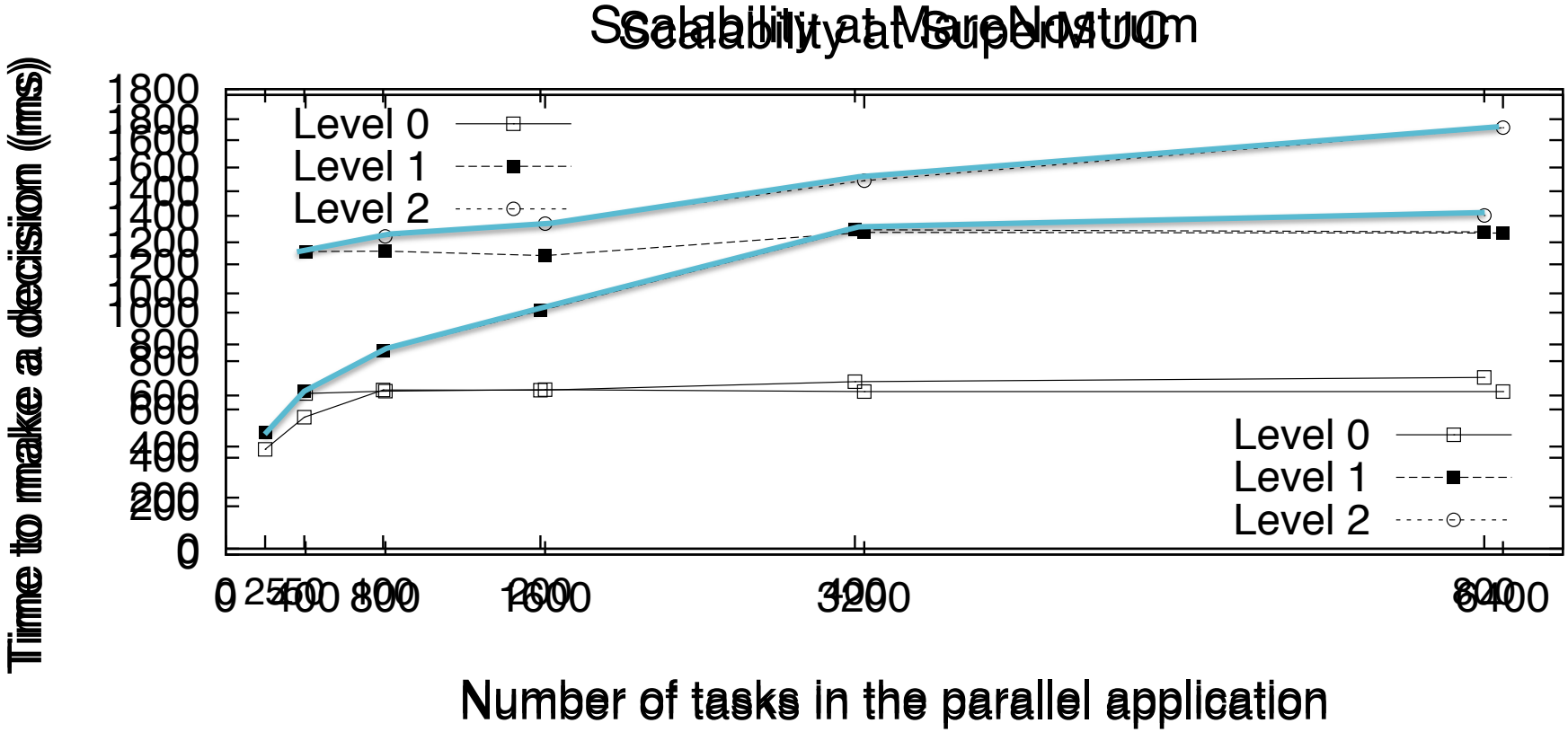
| # Tasks of the parallel application | Level 0 | | Level 1 | | Level 2 | |
|---|---|---|---|---|---|---|
| | #Analysis modules | Domain size | #Analysis modules | Domain size | #Analysis modules | Domain size |
| Architectures executed in MareNostrum and SuperMUC | | | | | | |
| 25 | 2 | 13 | 1 | 2 | - | - |
| 50 | 3 | 16 | 1 | 3 | - | - |
| 100 | 5 | 20 | 1 | 5 | - | - |
| 200 | 10 | 20 | 1 | 10 | - | - |
| 400 | 19 | 22 | 1 | 19 | - | - |
| 800 | 37 | 22 | 2 | 19 | 1 | 2 |
| Architectures executed in SuperMUC | | | | | | |
| 1600 | 73 | 22 | 4 | 20 | 1 | 4 |
| 3200 | 146 | 22 | 7 | 21 | 1 | 7 |
| 6400 | 292 | 22 | 14 | 21 | 1 | 14 |

# Scalability Evaluation

## Time to make a decision



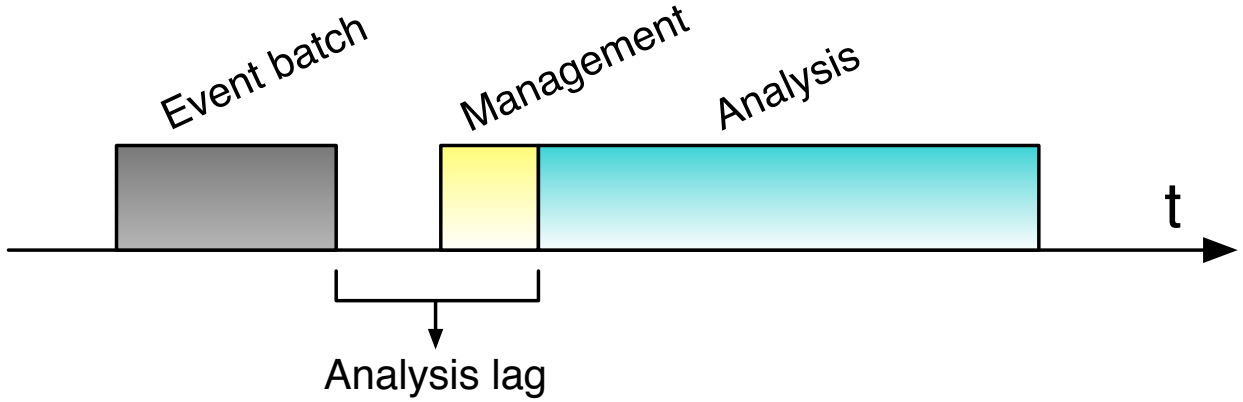
# Scalability Evaluation



# Scalability Evaluation

Efficiency of the hierarchical tuning network

## When is an analysis module saturated?




400 tasks

| Experiment | Level 0 #Analysis modules | Level 0 Domain size | Level 1 #Analysis modules | Level 1 Domain size |
|---|---|---|---|---|
| 1 | 19 | 21 | 1 | 19 |
| 2 | + 20 | 20 | 1 | 20 |
| 3 | - 18 | 22 | 1 | 18 |

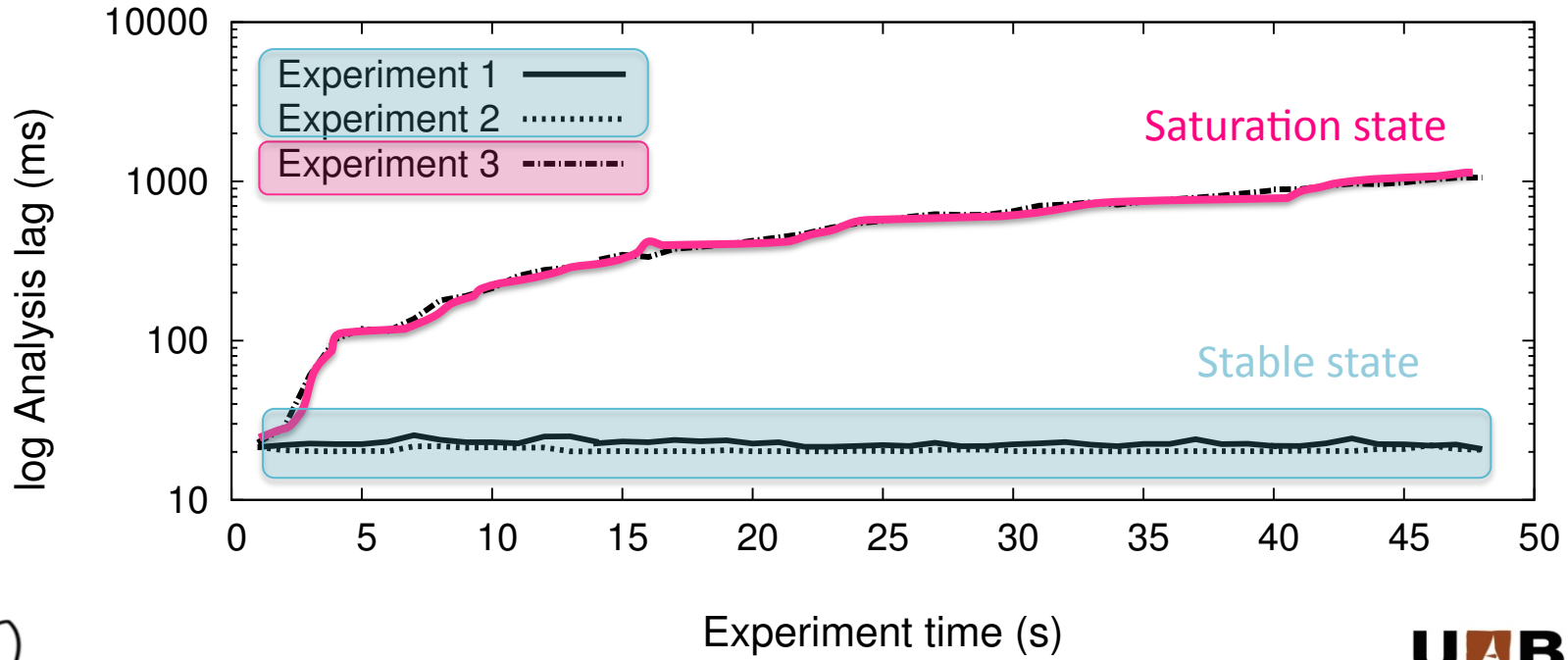
$$N \cdot E_a \cdot T_m + T_a(N) + \frac{E_a}{E_c} \cdot T_c + T_t \cdot f_{rp}$$

PREDICTED TOPOLOGY

# Scalability Evaluation

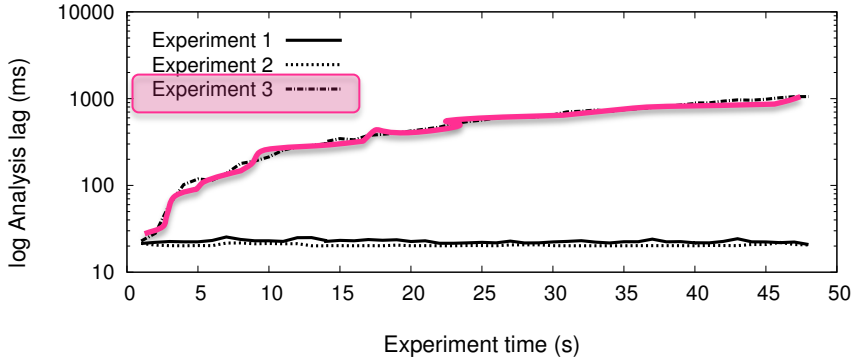
| Experiment | Level 0 #Analysis modules | Level 0 Domain size | Level 1 #Analysis modules | Level 1 Domain size |
|---|---|---|---|---|
| 1 | 19 | 21 | 1 | 19 |
| 2 | 20 | 20 | 1 | 20 |
| 3 | 18 | 22 | 1 | 18 |

Analysis lag of level 0 analysis modules

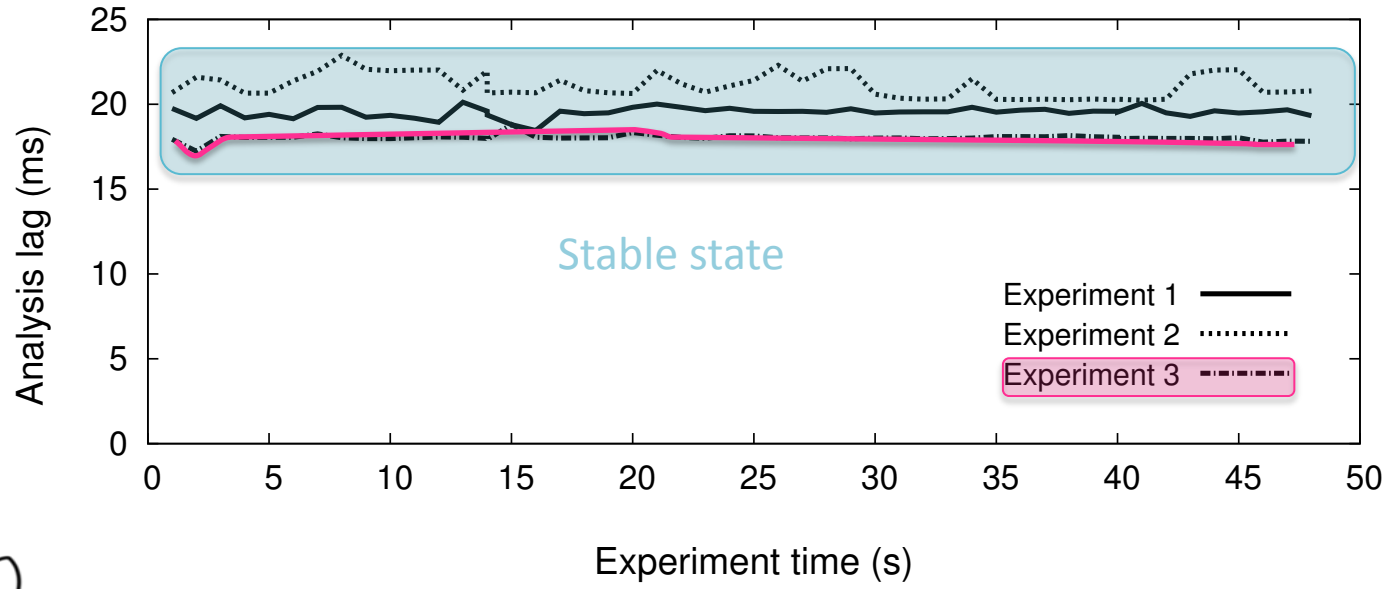


# Scalability Evaluation

| Experiment | Level 0 #Analysis modules | Level 0 Domain size | Level 1 #Analysis modules | Level 1 Domain size |
|---|---|---|---|---|
| 1 | 19 | 21 | 1 | 19 |
| 2 | 20 | 20 | 1 | 20 |
| 3 | 18 | 22 | 1 | 18 |




## Analysis lag of level 1 analysis modules



# Outline

- Motivation.
- Large-scale Dynamic Tuning.
- Scalability Evaluation.
- Conclusions and future work.

# Conclusions

- A model for distributed performance analysis based on a hierarchical tuning network has been defined.
- The decentralised decision making process employs user provided performance models and an abstraction mechanism.
- The scalability of the proposed model has been verified using a prototype of the tuning network.

# Future work

- Show the benefits of our approach when applied to real large-scale applications in order to improve their performance.
- Combine our approach with the one implemented under the AutoTune project.


HIPS 2013 Conference

# Thank you for your attention

amartinez@caos.uab.es

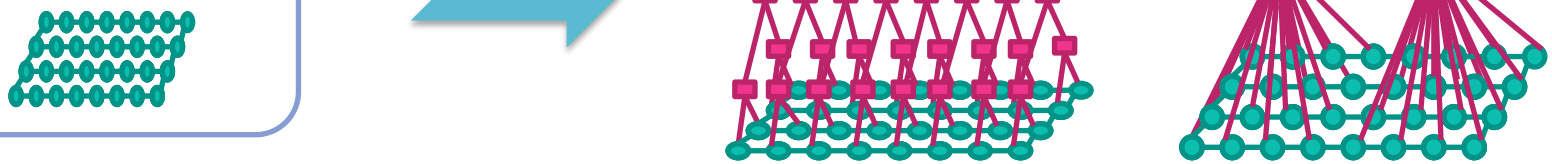
**Andrea Martínez, Anna Sikora, Eduardo César, Joan Sorribes**

Computer Architecture and Operating Systems Department
Universidad Autónoma de Barcelona

Universitat Autònoma de Barcelona

# Determining the architecture topology

Parallel application of N tasks



The use of architecture topologies with the minimum number of non-saturated analysis modules.

Work during the analysis and tuning process

$$N \cdot E_a \cdot T_m + T_a(N) + \frac{E_a}{E_c} \cdot T_c + T_t \cdot f_{rp}$$

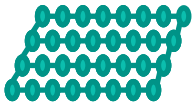
Gathering performance data

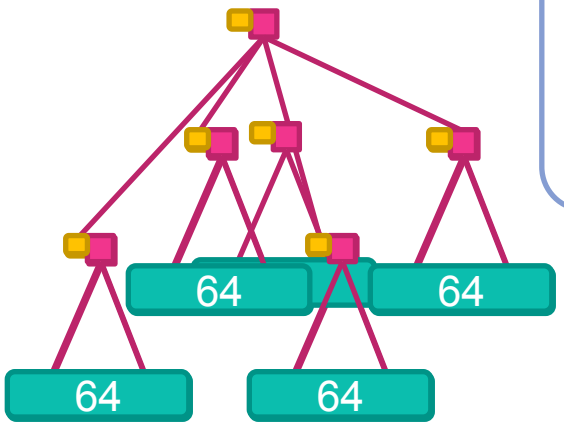
Performance analysis

Event creation

Order translation

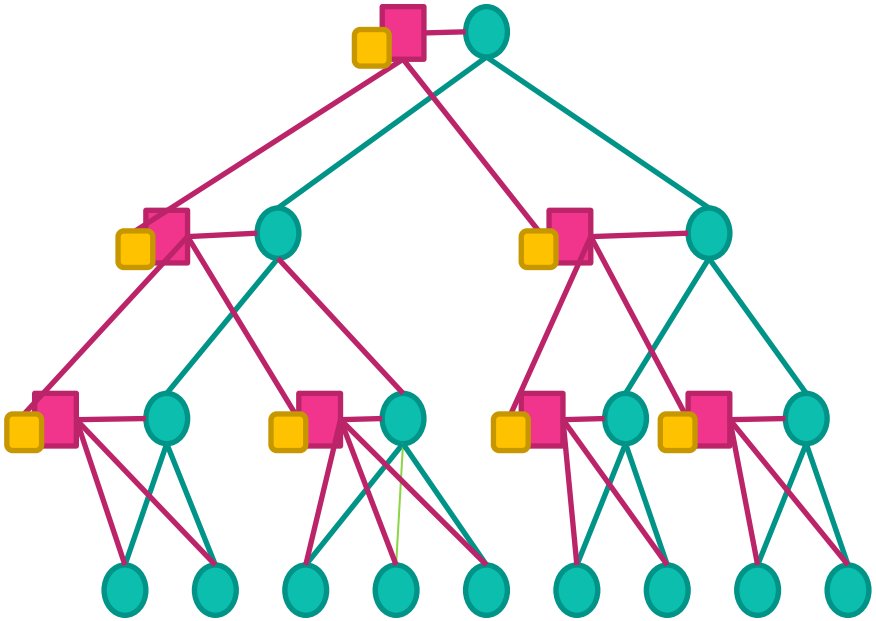
256 tasks
Parallel application





# Hierarchical Tuning Network

Hierarchical Master/Worker



Master/Worker of pipelines